Analytics Dashboards and User Behavior: Evidence from GitHub

Jan Schilpp*, Florian Pethig†, Hartmut Hoehle‡

January 10, 2025

Abstract

Open-source software (OSS) development is a social phenomenon, centered around connecting, collaborating, and comparing with other developers. To showcase their abilities and track their performance, developers maintain public user profiles that display detailed information about their OSS activities. In this paper, we examine how the adoption of an analytics dashboard that publicly displays performance information influences developers' contribution behavior—specifically, the quantity, effort, and diversity of their code contributions—as well as the developers' affect. Our empirical strategy is difference-in-differences using granular data from more than 60,000 GitHub developers over three years. The results indicate that the adoption of the analytics dashboard has heterogeneous effects on developers' behavior. High-activity developers benefit from the quantification and comparison of their work, becoming more active, particularly in collaborative projects. In contrast, low-activity developers exhibit different patterns. They increase their contributions mainly within the "safe space" of their own projects, but their commits are often accompanied by more negative messages. We present suggestive evidence that this increased negativity is linked to stress because it only occurs during peripheral and bothersome OSS activities, such as dependency handling. Our results show that public performance information may be a double-edged sword: although it can drive performance improvements, it may also impose psychological strain on developers. We discuss implications for theory and practice.

Keywords: analytics dashboard, open-source software, developer, user behavior
Appendix: The appendix is available at https://osf.io/ndy45?view_only=0b2450fa6fdd4d9981937b8651af0d5b

1 INTRODUCTION

Open-source software (OSS) is valued at eight trillion dollars globally, and organizations that engage active OSS developers can experience enhanced growth (Hoffmann et al. 2024, Wright et al. 2024). However, this may come at a cost to individual developers. 45% of OSS developers

*University of Mannheim, Germany, jan.schilpp@uni-mannheim.de
†Tilburg University, The Netherlands, f.pethig@tilburguniversity.edu
‡University of Mannheim, Germany, hoehle@uni-mannheim.de

indicate that maintaining OSS increases their level of personal stress (Tidelift 2023). Central to this phenomenon is the transparent nature of OSS, where the performance of individual developers is publicly visible. For example, on OSS development platforms like GitHub, the number of code contributions by more than 100 million OSS developers can be tracked and monitored by anyone. This issue is subject to intense debate in the OSS community. While some developers enjoy sharing their activities, others are concerned about the negative impact this transparency can have on their well-being, including an increased risk of burnout (Chan 2022).[1]

In this paper, we examine how publicly available performance information can shape social interactions among OSS developers. OSS development is a highly social phenomenon that revolves around connecting, collaborating, and comparing between developers. A key component of these interactions is personalized user profiles, where developers share information about themselves, such as their work history, project contributions, or personal interests (Marlow et al. 2013). Additionally, users rely on various extensions to showcase their performance, such as the *GitHub Readme Stats* analytics dashboard, a widely adopted user profile extension with more than 100,000 users. This dashboard visualizes detailed performance information about a user's activities on their profile, such as contribution quantity, and ranks developers relative to other GitHub developers using a letter-grade system.

The effect of the publicly-displayed performance information, such as that provided by the analytics dashboard, on developers' contribution behavior is unclear. On the one hand, the improved visibility of performance information may motivate developers to increase their contributions by encouraging them to outperform their peers (Chen et al. 2010, Landers et al. 2017). On the other hand, the comparison with peers may induce stress among some developers, especially when such comparisons with others are unfavorable (de Vries and Kühne 2015, Lee 2014, Wu et al. 2021). For example, in a discussion about the *GitHub Readme Stats* analytics dashboard a user notes "[o]f

course, almost nobody would like to understand he is not that good compared to others".[2]

Prior literature has focused on developers' motivation to contribute to OSS but does not provide deeper insights into the effect of public performance information on developers' contribution behavior. A large stream of literature has investigated why developers voluntarily contribute to OSS, emphasizing the importance of social interactions among developers. OSS developers derive satisfaction from being an integral part of a community, require value and ideology congruence when contributing to OSS, and are motivated by unique work structures in OSS development, such as autonomy or self-organization (Daniel et al. 2018, Maruping et al. 2019, Medappa and Srivastava 2019, von Krogh et al. 2012, Zhang et al. 2013). Furthermore, research indicates that providing people with performance information in online environments has positive effects, for example, on the number of contributions (Chen et al. 2010, Dissanayake et al. 2018, Dobrescu et al. 2021, Huang et al. 2019, 2021, Shi et al. 2021). However, much of the existing research focuses on privately-displayed performance information. The impact of public performance information may be different because public performance information, in particular in the OSS context, plays a crucial role in reputation building and facilitating comparison within the community (Dabbish et al. 2012, Hauff and Gousios 2015). Therefore, we investigate the following research question:

RQ: How does the adoption of an analytics dashboard that publicly displays OSS developers' performance influence their contribution behavior and affect?

In this study, contribution behavior refers to the number of developers' code contributions, as well as the associated effort and diversity. Because the analytics dashboard facilitates social comparison among developers, we draw from social comparison theory, which posits that individuals evaluate their own abilities by comparing themselves to others with greater abilities (Festinger 1954).

Empirically, we address this question using data from 63,135 GitHub developers over three

years, with one-sixth of them adopting the *GitHub Readme Stats* analytics dashboard. Our empirical strategy employs a difference-in-differences (DiD) approach. To mitigate endogeneity concerns arising from the voluntary adoption decision, we employ multiple validation strategies. Most importantly, we use synthetic DiD (synthDiD), a recent methodological advancement that reweights control units to ensure parallel pre-treatment trends, enabling valid identification in our context (cf., Arkhangelsky et al. 2021, Berman and Israeli 2022, Yeverechyahu et al. 2024). We complement this with additional approaches, including an instrumental variable estimation and a two-way fixed effects DiD using a matched sample of adopters and non-adopters.

Our findings provide robust evidence that adopting the analytics dashboard increases a developer's number of code contributions by up to 55%, as well as the associated effort and contribution diversity. Further analyses reveal that this effect differs between low- and high-activity users. High-activity users appear to benefit from the comparison with others, leading them to write more positive documentation in collaborative projects. Additionally, they disproportionately increase their number of contributions, effort, and diversity in these projects compared to low-activity users. In contrast, low-activity users primarily adjust their behavior within the "safe space" of their own projects and write more negative code documentation. We present suggestive evidence that the increased negativity may be due to stress, as it primarily occurs in response to bothersome, peripheral OSS activities such as dependency handling.

This study makes three important contributions to the research streams on OSS developers and performance information on online platforms. First, it extends existing research on the factors impacting OSS developers' contribution behavior by examining a previously unexplored factor: publicly-displayed performance information. Our findings show that developers' contribution behavior can be shaped not only by external influences but also by self-imposed interventions. Second, by applying social comparison theory to the field of OSS development, we highlight how the

complex social interactions among OSS developers can shape their contribution behavior. Third, our study finds that publicly-displayed performance information may be a double-edged sword, improving performance at the expense of potentially negative consequences for some developers. Therefore, developers and online platforms should carefully consider whether to publicly display performance information, especially for low-activity users.

2 BACKGROUND

The contribution behavior of OSS developers is a major topic in information systems (IS) research, investigating why people contribute to OSS and how interventions may impact their contributions. Our paper adds to this research stream by examining the impact of publicly-displayed performance information—specifically through an analytics dashboard on OSS developers' profile pages—on developers' contribution behavior and affect.

2.1 OSS Developers' Contribution Behavior

OSS is characterized by its accessibility, allowing anyone to freely access, modify, and distribute the software. As a result, it relies heavily on voluntary contributions by OSS developers. Platforms like GitHub offer the necessary infrastructure for efficient software development (e.g., version control) and communication among developers. Prior research highlights the role of social interactions within the OSS community in influencing developers' contribution behavior, a phenomenon known as "social coding" (Dabbish et al. 2012). Because OSS development is a concerted effort of many people who, mostly voluntarily, interact and collaborate to produce software, developers are often driven by intrinsic motivations. For example, developers are motivated by the unique work structures in OSS, which offer high levels of autonomy and self-organization (Medappa and Srivastava 2019, Lindberg et al. 2024). Similarly, values and ideology play a crucial role, as alignment between developers and their respective communities is an important predictor of developers' OSS contributions (Daniel et al. 2018, Maruping et al. 2019). Reputation and status of a user within the community are also

important elements of social coding on OSS development platforms (Dabbish et al. 2012, Hauff and Gousios 2015). Comparisons among community members can be influenced by performance information provided to the users, which may ultimately impact their contribution behavior.

2.2 Online Performance Information

Prior research finds that offering performance information in online environments has positive effects, for example, on the number of contributions. Table 1 presents an overview of related empirical studies. For example, Huang et al. (2019) reveal that offering performance information motivates users of a recipe crowdsourcing platform to contribute more frequently. Notably, performance feedback appears to be especially motivating for initially lower performing individuals (Chen et al. 2010, Huang et al. 2019).

Importantly, most existing studies examine the impact of privately-displayed performance information—information only visible to the user. Only Dissanayake et al. (2018) analyze the impact of publicly-displayed performance information and find that highly skilled teams exert more effort when they are close to winning an online innovation tournament. Yet, it is not clear whether the findings from this study generalize to OSS developers' contribution behavior in response to publicly-displayed information. First, Dissanayake et al.'s (2018) research is conducted at the team-level but team behavior could be inherently different from individual behavior due to the need for communication and collaboration in a team (Lu et al. 2012). Second, the online innovation tournament setting is time-bound, meaning that participants' behavior could be influenced by the short-term nature of the competition. In contrast, OSS developers are typically exposed to publicly-displayed performance information over a prolonged period, which may induce different behavioral outcomes such that initial positive responses may revert when someone is constantly exposed to the same stimulus (cf., Califf et al. 2020).

Furthermore, prior research on the impact of performance information visibility—whether

Table 1: Related Empirical Studies on the Effect of Performance Information in Online Environments

Study	Method	Unit of analysis	N	Environment	Performance information	Effect
	Privately-visible performance information					
Chen et al. (2010)	Field experiment	Individuals	398	Movie recommendation platform	Message with information on users' median contributions	Participants who initially contributed less than the median contribute more after treatment while the opposite is observed for participants with above-median initial contributions.
Huang et al. (2019)	Field experiment	Individuals	1,129	Recipe crowdsourcing platform	Messages with differently framed feedback	In terms of motivating user contributions, feedback notifications with a cooperative framing have the highest effect for females while notifications with a competitive framing work best for males.
Shi et al. (2021)	Field experiment	Individuals	7,046 & 7,150	Dating platform	Message with popularity information relative to other users	Users respond to unfavorable popularity feedback by increasing their self-marketing efforts, whereas favorable feedback leads to the opposite effect.
Dobrescu et al. (2021)	Randomized control trial	Individuals	1,101	Student online assignment	Interim ranking	Students exposed to the ranking information perform better and they engage in more peer discussions.
Huang et al. (2021)	Field experiment	Individuals	7,844	Massive online open course	Message about other students completion percentage	Students that received the information on other students' completion percentage show a higher completion probability and shorter completion time.
	Publicly-visible performance information					
Dissanayake et al. (2018)	Observational data	Teams	>10,000	Innovation tournament platform	Interim ranking	Teams closer to winning, in particular higher skilled ones, exert more effort toward the end of the tournament.
This study	Observational data	Individuals	63,135	OSS development platform	Analytics dashboard showing contribution metrics	Adopting the analytics dashboard increases a developer's number of contributions, the effort per contribution, and contribution diversity but users with low initial activity write more negative commit messages.

private or public—on individuals' reactions remains inconclusive, particularly in the OSS context. The literature suggests that when individuals are exposed to private performance information, they tend to compare themselves with those who outperform them. Conversely, when faced with public performance information, they may prefer to compare themselves with peers of similar or lower performance levels (Buunk and Gibbons 2007, Gibbons et al. 2002). Thus, the reference group for comparison may depend on the visibility of the performance information. Consequently, a person's behavioral response to the comparison might differ, especially since changes in behavior tend to be more noticeable when the comparison is unfavorable (cf., Chen et al. 2010).

Moreover, standing and reputation are key motivational factors for OSS community members and publicly-visible performance information alters how peers perceive the focal developer (Dabbish et al. 2012, Hauff and Gousios 2015, von Krogh et al. 2012). Given the importance of others' perceptions, public performance metrics may provide additional incentives for users to improve because it alters others' perception of the focal user. To our knowledge, no study has examined how publicly-displayed performance information, such as an analytics dashboard on a developer's profile page, affects OSS developers' contribution behavior. Given the importance of social interactions in the OSS community, as well as the comparison opportunities enabled by the analytics dashboard, we draw from social comparison theory. This approach helps us better understand the interplay between publicly-displayed performance information and user behavior.

3 THEORY AND HYPOTHESES DEVELOPMENT

Social comparison theory posits that individuals have the desire to evaluate themselves to determine whether they perform well or poorly (Festinger 1954). As objective standards are often unavailable, individuals evaluate themselves in relation to others. There is abundant research that supports the notion that individuals tend to choose upward comparison and compare themselves to those who are slightly more skilled or accomplished (Gerber et al. 2018). There are two main

reasons for this strategy. First, to find an appropriate comparison group, individuals tend to select similar peers because such comparisons promise a more accurate self-assessment of their abilities. Second, upward social comparison is deeply rooted in our culture that values higher performance (Festinger 1954, White and Lehman 2005).

Prior research identifies social comparison as an important driver of users' behavior and affect in the online environment broadly, and in the OSS development field specifically. For example, adverse social comparison on social media may lead to negative affect as people become envious (Krasnova et al. 2015). Simultaneously, there is preliminary evidence that adverse social comparison motivates OSS developers to improve their contributions after benchmarking against other OSS projects (Lumbard et al. 2024).

In this realm, OSS developers' contributions take many forms. In our study, we focus on code contributions because additions or changes to the code are the primary objectives in software development. Here, we investigate how adopting the analytics dashboard influences developers' quantity, effort, and diversity of code contributions. Furthermore, we hypothesize how developers' affect is impacted to assess the sustainability of the intervention with respect to developers' well-being (cf., Matook et al. 2021).

The analytics dashboard publicly displays performance information about OSS developers' activities on their user profiles, such as the number of code contributions, and ranks developers relative to all other GitHub developers using a letter-grade system. This visibility allows adopters to easily compare their performance with peers, either by observing their rank or inspecting detailed statistics relative to other adopters, such as collaborators. According to social comparison theory, we expect adopters to leverage this information to compare themselves to higher performing others, motivating them to improve their performance by narrowing the gap between themselves and those to whom they compare themselves (Burtch et al. 2018, Chen et al. 2010, Festinger 1954, Landers

et al. 2017). Thus, upward social comparison facilitated by the analytics dashboard is expected to motivate developers to improve their number of contributions.

However, the magnitude of this effect may be contingent on developers' initial activity levels, which are known to follow a power law distribution—where a small number of users account for the majority of contributions (Johnson et al. 2014, Kalliamvakou et al. 2016). While high-activity users constitute experienced and well-established OSS developers, low-activity users are inexperienced or occasional contributors. This distinction is important for two reasons. First, low-activity users have a higher probability of engaging in upward social comparison because they have a larger pool of higher-performing developers to compare themselves to. Second, explaining personally adverse social comparison is a known coping mechanism that may diminish the behavioral effect that follows upward social comparison (Buunk and Gibbons 2007). However, low-activity users may lack the background knowledge to contextualize or explain other developers' higher number of code contributions. For example, they may not recognize that another developer's greater output could be due to lower effort per contribution. Experienced developers, by contrast, should be able to quickly identify and explain such nuances.

Taken together, we expect all developers to increase their number of code contributions due to heightened motivation stemming from upward social comparison facilitated by the analytics dashboard, but anticipate this effect to be stronger for low-activity users, as they are more likely to engage in upward social comparison and may lack coping mechanisms. Thus, we hypothesize:

H1: *Developers' number of code contributions increases after adopting the analytics dashboard. This effect is stronger for developers with low initial activity.*

Moreover, people strive to maximize benefits while minimizing costs. For comparisons on online platforms, this means that users want to achieve the highest possible rank with minimal effort, strategiccally allocating their resources (Dissanayake et al. 2018). As software developers'

OSS platform profiles are increasingly relevant in recruiting processes (Hauff and Gousios 2015), developers may use the analytics dashboard to transmit a positive image of themselves. In line with social comparison theory, we therefore expect adopters to allocate their effort on increasing the total number of code contributions displayed in the analytics dashboard, while reducing the effort invested in each individual contribution. Following our previous arguments, we assume that the effort reduction is more pronounced for developers with initially lower activity.

Put differently, our reasoning suggests that developers may act strategically by increasing the number of contributions, even at the expense of the effort put into a single contribution. In doing so, they aim to improve the performance image presented by the analytics dashboard, which should be especially relevant for low-activity users having initially lower performance metrics. Thus, we hypothesize:

H2: *Developers' effort per code contribution decreases after adopting the analytics dashboard. This effect is stronger for developers with low initial activity.*

The cost-benefit calculations of developers who adopt the analytics dashboard and aim to match the number of contributions of more active peers at the lowest possible costs may lead not only to a reduction in effort per contribution but also to a decrease in contribution diversity. Contributions to OSS take many forms, and given the mostly voluntary nature of OSS development, developers can choose the type of code contributions they make. Examples include optimizing existing software code—such as making it more computationally efficient or fixing bugs—adding new features, or engaging in management-like tasks like reviewing code contributions suggested by other developers (Hattori and Lanza 2008). Each of these activities requires different skill sets and prior knowledge. For example, adequately optimizing current code demands a deep understanding of the existing codebase, while adding new features requires creativity and a comprehensive overview of the entire software.

Engaging in a diverse set of activities forces a developer to familiarize themselves with many aspects of the project and its code, each associated with potentially high initial costs. For instance, a developer who has focused on adding new features would need to gain detailed knowledge of the existing code before engaging in optimization tasks like bug fixing. This familiarization consumes resources that the developer could have allocated toward contributing within their "comfort zone," potentially resulting in a higher number of contributions. This effect is likely to be particularly pronounced for low-activity users who lack experience from other projects, thereby disproportionally increasing the setup costs associated with more diverse types of contributions.

So, we expect developers to focus on a few development activities they excel at, allowing them to maximize their efforts in improving the numbers displayed on the analytics dashboard. This pattern should be especially pronounced among low-activity developers, as they face disproportionately higher setup costs when engaging in diverse OSS activities. Based on these arguments, we hypothesize:

H3: *Developers' code contribution diversity decreases after adopting the analytics dashboard. This effect is stronger for developers with low initial activity.*

Lastly, frequent upward social comparisons can have negative effects on developers' affect when others are perceived as superior (de Vries and Kühne 2015, Krasnova et al. 2015, Lee 2014, Yue et al. 2022). Stress, in particular, has been identified as a significant problem among OSS developers (Raman et al. 2020). Califf et al. (2020) conceptualize stress as a process that includes (1) stressors (e.g., interpersonal expectations) that induce (2) positive or negative psychological responses, which in turn lead to (3) outcomes in the form of a positive or negative psychological state. The negative psychological state could be stress, commonly defined as the feeling that arises when an individual lacks sufficient resources to satisfy external demands (Amirkhan et al. 2018).

While upward social comparison can lead to positive psychological responses and outcomes

by motivating individuals to match those perceived as better—for example, by increasing the code contribution quantity—the initial positive response to a stressor can revert if it is excessively triggered (Califf et al. 2020). Thus, as developers excessively compare themselves with others due to the analytics dashboard's prominent placement on their central profile page, this may lead to negative outcomes for adopters. This effect may be contingent on a developer's initial activity level on the platform; low-activity users are more likely to engage in adverse social comparison and may lack the knowledge to fully explain discrepancies in performance metrics.

Based on these arguments, we expect rising stress levels among analytics dashboard adopters due to frequent unfavorable social comparisons. This effect is expected to be stronger for low-activity users, as they are more likely to engage in upward social comparison and may lack coping mechanisms. Thus, we hypothesize:

H4: *Developers' stress increases after adopting the analytics dashboard. This effect is stronger for developers with low initial activity.*

4 SETTING

We test our hypotheses in the context of GitHub, the most popular platform for OSS development with more than 100 million users (GitHub Inc. 2023). On GitHub, software code contributions are labeled as *commits*, which are changes to one or multiple files within a project, known as a *repository*. Along with commits, developers provide commit messages in which they document the modifications. This documentation is crucial in OSS development because it facilitates understanding of the code and tracking of changes over time. Furthermore, GitHub offers social features such as user profiles and the possibility to follow other users.

This paper exploits the adoption of the *GitHub Readme Stats* analytics dashboard by many GitHub developers on their publicly accessible profile pages (Hazra 2020). The dashboard is depicted in Figure 1. It displays descriptive statistics related to a user's contributions on GitHub,

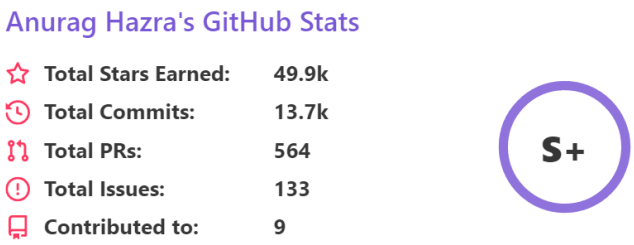

Figure 1: Analytics Dashboard.

Note. Retrieved from https://github.com/anuraghazra/github-readme-stats (October 11, 2022).

such as their number of commits (*Total Commits*), pull requests (proposed code changes, *Total PRs*), and issues (discussion threads, *Total Issues*). Furthermore, the dashboard documents the number of stars the adopters' repositories have received (*Total Stars Earned*). Users can star a repository to show appreciation for the work (GitHub Inc. 2022). The dashboard also shows the unique number of repositories to which the adopter has contributed (*Contributed to*). Based on these metrics, a rank is calculated that categorizes the adopter relative to all other GitHub users using a cumulative distribution function. The ranks range from B+ (top 100%) to S+ (top 1%).

5 METHODOLOGY

5.1 Data

The analytics dashboard was initially launched on July 9, 2020. We identified 137,012 adopters of the analytics dashboard within a two-year window between July 9, 2020, and July 9, 2022. Furthermore, we identified the GitHub users whom the adopters were following and selected a random sample of 300,000 non-adopters from this group of followees. We then collected all public GitHub activities of these 437,012 users between January 22, 2020, and December 25, 2022, from GhArchive (for details, see Appendix A).

We operationalize our dependent variables as follows. The code contribution quantity is measured by a user's number of commits (*NumCommits*). We approximate the effort per contribution by measuring the commit message length (*MesLength*), calculated as the number of characters per

commit message[3]. As outlined, commit messages contain descriptions of the software modifications and are considered highly important by developers. Thus, if a commit required greater effort, this is likely reflected in a longer commit message. A developer's contribution diversity is measured by the number of unique words in their commit messages (*MesUniqueWords*). Finally, We exploit that the messages not only contain information about the commit itself, but also meta-information in the form of the commit message sentiment (*MesSentiment*), which has been found to reflect the developer's affect (Wu et al. 2021). We determine the sentiment using a BERT (Bidirectional Encoder Representations from Transformers) model (Devlin et al. 2019) that is fine tuned to the software engineering domain. The model classifies a commit message sentiment on a scale from -1 (negative) to 1 (positive). Table 2 presents exemplary negative, neutral, and positive commit messages. For more details on the BERT model refer to Appendix B.

Table 2: Exemplary Commit Messages

Commit message	Sentiment
Tried everything, still haven't figured it out...	−0.999
Still can't solve :cry:	−0.998
Fix total objects if offset is higher than total	0
Merge pull request No. 40 from vikrambombhi/update-uml. update UML to allow multiple answers	0
Great. Everything seems working.	0.999
Handle parallelize typing issues. And move it to Python 3. Yay!	0.999

Note. The exemplary messages are from March 18, 2020 and are classified by the fine tuned BERT model. The sentiment score ranges from -1 (negative) to 1 (positive).

Table 3: Summary Statistics

Variable	N	Mean	SD	Median	Min	Max
NumCommits	7,073,237	15.922	69.860	0	0	8,354
MesLength	2,888,879	49.395	133.771	28	0	35,758.364
MesUniqueWords	2,892,877	108.209	345.774	28	0	32,762
MesSentiment	2,888,653	−0.001	0.089	0	−0.999	0.999

Note. The summary statistics displayed here are based on all user-month observations eligible for the synthDiD. The variables *MesLength* and *MesSentiment* are mean values per user and month. The number of observations varies between the variables because developers may commit without including a message, or the message may be too short to compute the *MesLength*, *MesUniqueWords*, or *MesSentiment*, which can lead to missing data.

Finally, we aggregate the number of commits (count), the commit message length (mean), the number of unique words in commit messages (count), and the commit message sentiment (mean) on a monthly level, i.e., 28-day periods relative to the initial analytics dashboard launch on July 9, 2020. This was done to make the analysis computationally feasible and reduce noise in the data caused by irregular contribution behavior per day and week (cf., Guzman et al. 2014). As a result, we obtained a panel data set with one observation per user per month. Table 3 displays the summary statistics of the resulting variables. To account for skewness, we logged the number of commits, the commit message length, and the number of unique words for further analysis.

5.2 Identification Strategy

We use a DiD approach to compare the behavior of analytics dashboard adopters with that of non-adopters. Any additional difference in the post-treatment period relative to the pre-treatment period represents the treatment effect—that is, the effect of adopting the analytics dashboard (Angrist and Pischke 2008). Traditional DiD analysis imposes a strict parallel pre-treatment trend assumption, meaning the behavior of treated (adopters) and control (non-adopters) users must follow common trends before the treatment. In our case, the challenge lies in the voluntary nature of the adoption decision, which introduces the risk of endogeneity. Analytics dashboard adopters may change their behavior before adopting. For example, a developer seeking for a new job might prove their skills by contributing more on GitHub, which simultaneously increases the probability of becoming aware of and adopting the dashboard to showcase their increased contributions. This scenario complicates the clear identification of the actual effect of the analytics dashboard adoption on developers' contribution behavior because the motives of adopters may diverge from those of non-adopters.

To address this issue, we deploy synthDiD, a recent methodological advancement that allows for valid identification of treatment effects in settings where the parallel pre-treatment trend

assumption may be violated (Arkhangelsky et al. 2021). SynthDiD combines DiD and the synthetic control method by reweighting control units to achieve parallel pre-treatment trends. Consequently, any additional post-treatment deviation can be validly attributed to the treatment itself. However, the standard synthDiD approach does not account for the staggered adoption of the treatment, which is present in our setting that encompasses 38 months (January 23, 2020 to December 21, 2022) with users adopting the dashboard during any of 26 months (July 9, 2020 to July 6, 2022).

Therefore, we draw on Berman and Israeli (2022), who adapted synthDiD to handle staggered adoption. The authors developed this approach for a setting very similar to ours, allowing them to estimate the monetary benefits companies achieve after voluntarily adopting a descriptive dashboard. Their data comprised a company-month panel where some companies adopted the dashboard in any given month, requiring a DiD estimator capable of achieving parallel pre-treatment trends in a setting of voluntary staggered treatment. In Berman and Israeli (2022), this is accomplished by computing a separate synthDiD for every possible adoption period and aggregating the results to obtain an overall estimate. Each synthDiD encompasses six pre- and six post-treatment periods, with the adoption month being part of the post-treatment period.

The synthDiD modified for staggered adoption is computationally intensive because the algorithm must determine the optimal weights for the control units for every adoption period, which becomes increasingly complex for a larger number of units. To make this computationally feasible with our data encompassing 437,012 users, we adapted Berman and Israeli's approach. Specifically, our model did not converge when using all non-adopters as control units. Therefore, for every adoption month, we randomly selected a sample of non-adopters equal to five times the number of adopters in that adoption month. We ran the synthDiD for this adoption period with all relevant adopters as treatment group and the sampled non-adopters as control group. To avoid bias from the random selection of non-adopters, we repeated the process ten times, each time with another

random sample of non-adopters.[4] We proceeded by running this analysis for every adoption month and aggregated the results following Berman and Israeli.

Furthermore, to differentiate between low- and high-activity users, we performed a median split based on the adopters' total number of commits in the pre-treatment period. Since the number of commits is not uniform throughout our observation period, the cutoff point is dynamically calculated for each adoption month (for details see Appendix C). Eventually, we adhered to the time frame proposed by Berman and Israeli (2022) and analyze data from six months before and after the adoption (including the adoption month) to investigate the lasting impact of the analytics dashboard. The approach is summarized in Figure 2.

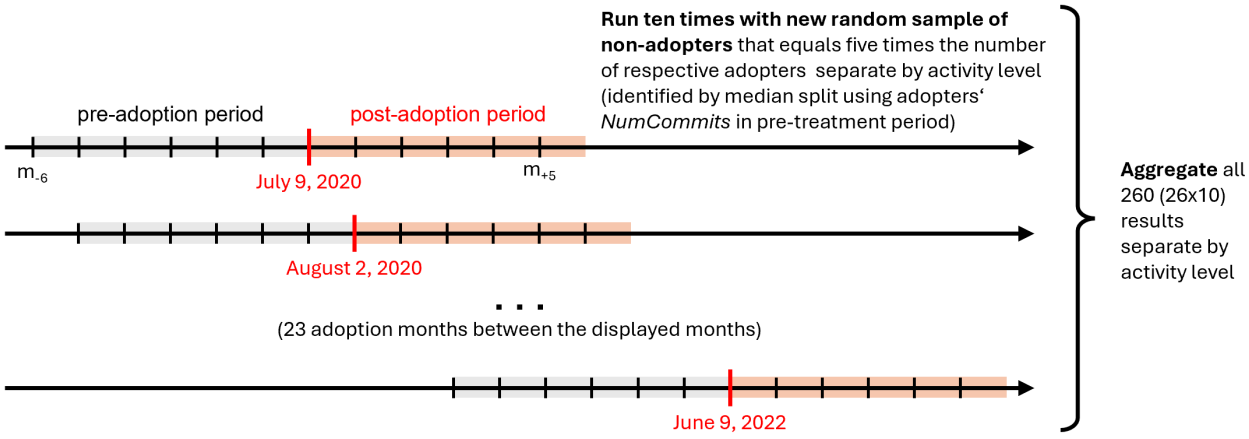

Figure 2: Modified Synthetic Difference-in-Differences Approach (based on Berman and Israeli 2022)

The following equation displays our regression specification in a simplified linear version (for the complete one refer to Appendix C):

$$y_{it} = \beta_0 + \beta_1 Adoption_i x After_{it} + u_i + \tau_t + \epsilon_{it} \tag{1}$$

where the dependent variable y_{it} denotes user i's logged number of commits (*NumCommits*, number of code contributions, H1), logged mean commit message length (*MesLength*, effort per code contribution, H2), logged number of unique words in commit messages (*MesUniqueWords*, code contribution diversity, H3), or mean commit message sentiment (*MesSentiment*, developer

stress, H4) in calendar month t. β_0 represents the constant of the regression. The variable $Adoption_i$ is a dummy indicating whether user i eventually adopts the dashboard (1 = adopter group) and $After_{it}$ indicates whether the observation month is part of the pre- or post-adoption period (1 = post-adoption). The interaction $Adoption_i x After_{it}$ is our DiD term. Thus, β_1 is our DiD estimator showing how adopting the analytics dashboard influences the respective dependent variable. Eventually, the number of commits are highly user- and time-dependent (Guzman et al. 2014, Kalliamvakou et al. 2016). Therefore, our specification accounts for user and time fixed effects, u_i and τ_t. ϵ_{it} represents the error term.

6 RESULTS

6.1 Main Results

Table 4 presents the regression results split by low- and high-activity users. Columns 1 and 2 show the effect of adoption on users' number of commits that we hypothesized to increase especially for low-activity users (H1). The DiD coefficient (*Adoption x After*) is positive and significant ($p < 0.01$) for both low- and high-activity users, indicating that both groups commit more after adopting the analytics dashboard. The synthDiD coefficients can be interpreted similarly to results from ordinary least squares (OLS) regressions. Since our dependent variable is logged, the coefficients roughly represent percentage changes. These findings suggest that adopters with low initial activity commit approximately 55% more after adoption, while adopters with high initial activity commit about 25% more. These findings are in line with our first hypothesis.

Columns 3 and 4 present the regression results of the analytics dashboard adoption on the mean commit message length. We anticipated this variable to decrease (H2) but contrary to our expectation both DiD estimators are positive and significant ($p < 0.01$), indicating that users write longer commit messages after adopting the analytics dashboard. Specifically, the mean number of characters for adopters with low initial activity increases by approximately 5%, while for high

activity users it increases by about 2% per message.

Next, we examine the impact of adopting the analytics dashboard on the unique number of words in the users' commit messages. The results of the corresponding synthDiD regressions are presented in columns 5 and 6. Again, we hypothesized a negative effect (H3), however, both DiD coefficients are positive and significant ($p < 0.01$) but they vary greatly in magnitude. Users with low initial activity increase the number of unique words by about 19%, whereas high-activity users show an increase of roughly 2%.

The effect on the commit message sentiment is presented in the columns 7 and 8. Here, we expected a negative treatment effect mirroring increased stress, in particular for low-activity developers (H4). While the DiD estimator for users with high initial activity is not statistically significant, the estimator for low-activity users is negative and significant, with a point estimate of -0.001 ($p < 0.01$). This indicates that low-activity users write more negative commit messages after adopting the analytics dashboard partially supporting our fourth hypothesis.

Table 4: Synthetic Difference-in-Differences

	NumCommits		*MesLength*		*MesUniqueWords*		*MesSentiment*	
	(1) Low	(2) High	(3) Low	(4) High	(5) Low	(6) High	(7) Low	(8) High
Adoption x After	0.550***	0.245***	0.051***	0.017***	0.193***	0.021***	−0.001***	0.000
	(0.004)	(0.005)	(0.002)	(0.002)	(0.005)	(0.004)	(0.000)	(0.000)
Observations	376,704	380,916	241,416	242,424	245,520	247,032	241,416	242,424
Users	31,392	31,743	20,118	20,202	20,460	20,586	20,118	20,202
Mean Adopters Pre	1.762	39.281	38.023	57.710	61.867	308.500	0.000	−0.002

Note. The dependent variables are displayed in the first row. Activity levels are displayed in the second row. Users are defined as high-activity if the number of commits during the pre-treatment period is equal to or above the sample median. The pre-treatment means of adopters for *NumCommits*, *MesLength*, and *MesUniqueWords* are presented in their raw form (not logged). The number of users and observations varies between models because synthDiD requires a balanced panel. Developers may commit without including a message, or the message may be too short to compute the *MesLength*, *MesUniqueWords*, or *MesSentiment*, which can lead to missing data. Robust standard errors in parentheses. *** $p < 0.01$, ** $p < 0.05$, * $p < 0.1$.

Figure 3 plots the dynamic treatment effect. Specifically, the plots show the point estimates and 95% confidence intervals (CI) of the regression coefficient *Adoption x After* for each time period around the adoption month. For all specifications, synthDiD achieves a very good pre-

treatment fit between adopters and non-adopters, evidenced by the few and barely significant treatment coefficients in the pre-treatment periods. Conversely, significant treatment effects in the post-treatment periods, consistent with the aggregated effects in Table 4, are visible.

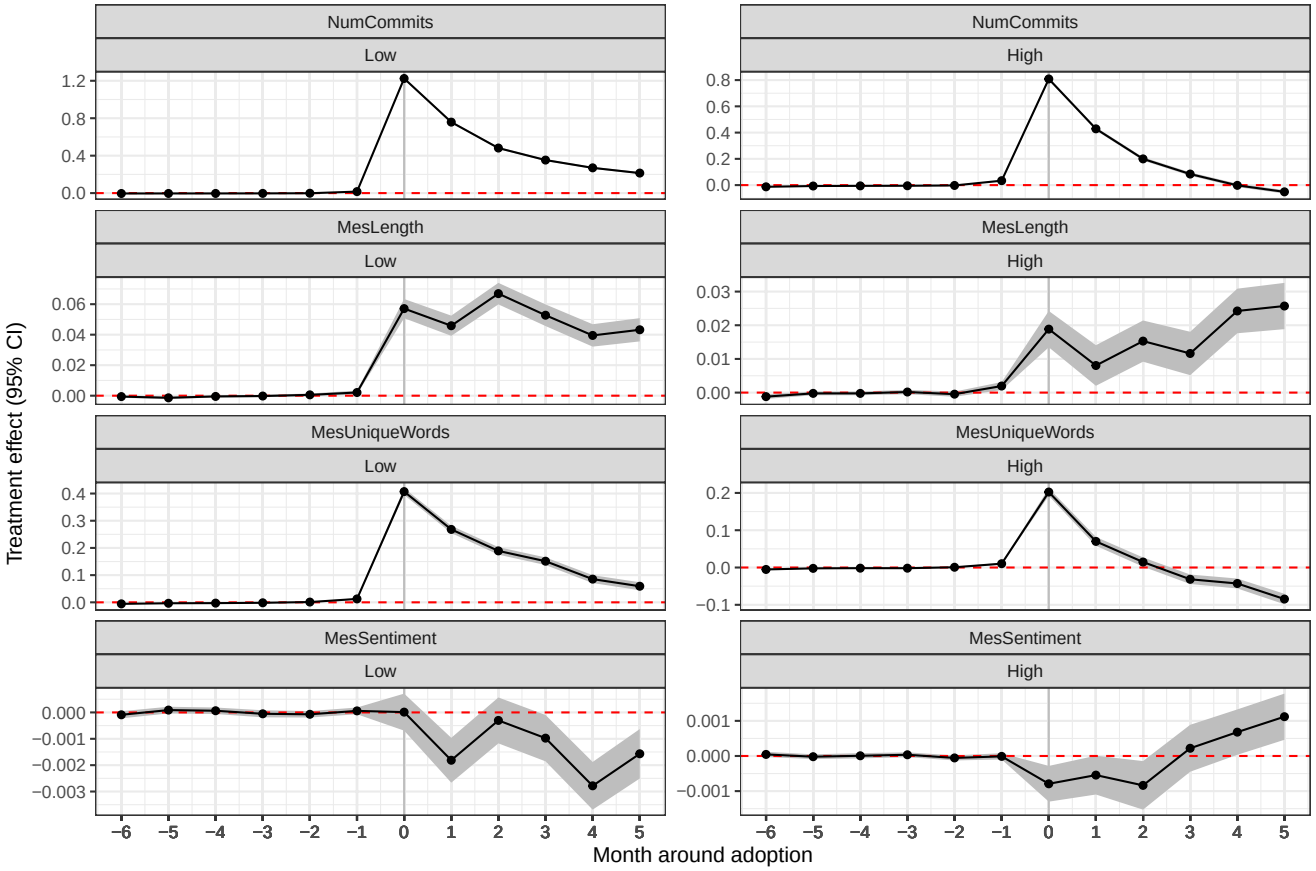

Figure 3: Synthetic Difference-in-Differences Event Studies

Note. The dependent variables are displayed in the first row. Activity levels are displayed in the second row. Users are defined as high-activity if the number of commits during the pre-treatment period is equal to or above the sample median.

First, the number of commits by adopters sharply increases in the treatment month, with the effect decaying over time for both low- and high-activity users. However, while there is a positive and significant treatment effect for low-activity users throughout all post-treatment periods, a slightly negative and significant treatment coefficient is observed in the last post-treatment period for high activity users. Second, the analytics dashboard adoption shows a positive effect on commit message length for both groups, which remains constant and significant throughout the post-treatment periods. Third, the event study plots for the number of unique words show a similar

pattern to that of the number of commits. There is a sharp increase in the number of unique words during the first adoption month, which decays over time and even turns negative during the last three post-treatment months for high activity users. Fourth, for commit message sentiment, we observe patterns that align with the aggregated regression results reported in Table 4: low-activity users write significantly more negative commit messages in four out of six post-treatment months, while the results are mixed for high-activity users, resulting in an insignificant overall effect.

6.2 Instrumental Variable Approach

As previously outlined, our data is susceptible to endogeneity issues, primarily stemming from potential simultaneity and omitted variable biases. Simultaneity bias arises from the reciprocal influence between the dependent and independent variable. Users who actively contribute on GitHub are more likely to become aware of and adopt the analytics dashboard. Consequently, there is a correlation between a user's number of contributions and the decision to adopt the analytics dashboard. Additionally, omitted variables—unobservable in our analysis—may also influence both adoption and contribution behavior. For instance, factors such as job-seeking, which may motivate OSS developers' to showcase their skills by contributing more on GitHub, could affect both their contribution behavior and their likelihood of adopting the dashboard.

To address these concerns, we employ an instrumental variable (IV) approach to account for the potential endogeneity of analytics dashboard adoption. This approach uses IVs that are correlated with the regressor (i.e., analytics dashboard adoption), but not with other confounding factors that could impact our dependent variables (e.g., increased motivation to use GitHub). We instrument the adoption of the analytics dashboard with the users' exposure to it, i.e., the probability that OSS developers become aware of the dashboard, thereby increasing the likelihood of adoption. This exposure comprises two components. First, the general attention the analytics dashboard receives on GitHub affects a user's likelihood of discovering it, independent of their own contribution

behavior. We measure this attention using monthly number of interactions with the analytics dashboard repository, such as comments, commits, forks (copies of the repository), and stars. Second, we leverage exogenous shocks that impact the visibility of the dashboard. For example, the analytics dashboard was occasionally featured on GitHub Trending, a daily list of popular repositories, and on various news websites and blogs. Conversely, there were periods when the dashboard was temporarily unavailable due to technical issues, reducing its visibility on adopters' profiles. We quantify the monthly impact of these shocks by calculating the number of affected days per month. These two types of IVs satisfy the requirements of exogeneity and relevance, making them suitable instruments for estimating the impact of the dashboard adoption despite the risk of simultaneity and omitted variable biases. Appendix D provides more details on our IV approach, outlining the exogenous shocks and their instrumental relevance.

We leverage these variables in IV regressions to accurately estimate the effect of analytics dashboard adoption on our dependent variables. Our analysis covers data from the two-year period following the creation of the analytics dashboard repository on July 9, 2020. We consider only the time period when the analytics dashboard was available to ensure that the IVs are measurable. Additionally, we restrict our sample to adopters with complete observations from six months prior to their adoption up to the month of adoption. This limitation is necessary because the IVs only instrument the adoption event, not the continuous use of the dashboard. We also control for user, relative month, and calendar month fixed effects. Our analysis includes 55,011 dashboard adopters.

Because this analysis focuses exclusively on adopters of the analytics dashboard, the primary variable of interest is *After (Instrumented)*, a dummy equal to one in the user's adoption month, which is instrumented by the variables described earlier. The coefficients associated with this variable provide insights into the impact of adopting the analytics dashboard on user behavior in the adoption month while accounting for the potential endogeneity bias.

The results of our analysis are reported in Table 5. The coefficients show the same direction and significance as in our main specification. Specifically, we find a positive impact on number of commits, commit message length, and number of unique words, as well as a negative impact on the commit message sentiment of low-activity users (columns 1-7, $p < 0.01$). The effect on commit message sentiment of high-activity users is not statistically significant (column 8). The magnitudes of the treatment effects are larger than those using our synthDiD specification, as the IV approach only estimates the treatment effect during the adoption month, while our synthDiD estimates the cumulative effect over six months post-adoption.

Table 5: Instrumental Variable Regression

	NumCommits		*MesLength*		*MesUniqueWords*		*MesSentiment*	
	(1) Low	(2) High	(3) Low	(4) High	(5) Low	(6) High	(7) Low	(8) High
After (Instrumented)	1.685***	1.427***	0.083***	0.102***	0.806***	0.533***	−0.005***	−0.001
	(0.016)	(0.023)	(0.012)	(0.008)	(0.020)	(0.017)	(0.002)	(0.001)
Observations	163,722	166,344	55,023	130,437	55,100	130,780	55,023	130,429
Users	27,287	27,724	24,339	27,687	24,361	27,693	24,339	27,684

Note. The dependent variables are displayed in the first row. Activity levels are displayed in the second row. Users are defined as high-activity if the number of commits during the pre-treatment period is equal to or above the sample median. The adoption is instrumented by the number of interactions, forks, and stars of the analytics dashboard repository, as well as the number of days it was featured or down. The number of users and observations vary between the models because due to missing values. Developers can commit without a commit message or the latter may be too short to compute the *MesLength*, the of *MesUniqueWords*, or the *MesSentiment* partly leading to missing observations. Robust standard errors in parentheses. *** $p < 0.01$, ** $p < 0.05$, * $p < 0.1$.

6.3 Additional Robustness Checks

We conduct several robustness checks. Most importantly, we estimate the treatment effect using a two-way fixed effects model on a matched sample of adopters and non-adopters accounting for user and time fixed effects. There, we formally test whether the DiD estimators in the regressions between low- and high-activity users significantly vary leveraging an approach outlined by Oberfichtner and Tauchmann (2021). The results from these analyses corroborate the findings from our synthDiD and instrumental variable specification (see Appendix E).

To conclude, our findings mostly support H1 (number of code contributions) and H4 (developer stress), indicating that the adoption of the analytics dashboard has a stronger effect on users with

low initial activity. Specifically, low-activity adopters commit about 30 percentage points more and write more negative commit messages after adoption, compared to high-activity adopters. Contrary to expectations, we reject H2 (effort per code contribution) and H3 (code contribution diversity), as the adoption unexpectedly leads to longer commit message and a higher number of unique words. These effects are stronger for low-activity users, suggesting they invest more effort and show greater diversity in their contributions than high-activity users. We explore these effects further through additional analyses based on social comparison theory.

6.4 Additional Analyses

We argue that the adoption of the analytics dashboard enables social comparison for adopters. We suspect that upward social comparison is driving the change in developers' contribution behavior. Furthermore, we argue that this effect is stronger for low-activity users, who may lack the experience to cope with adverse comparisons. By contrast, high-activity developers may be better equipped to navigate the OSS environment.

OSS development on GitHub occurs across many different repositories, each is owned by a distinct user. The repository owner plays a central role in managing the repository, which influences how contributors interact with and perceive the environment. For example, prior research has shown that the productivity of external contributors increases when repository owners react positively and provide timely feedback (Smirnova et al. 2022). When users contribute to repositories they own, they operate in a familiar environment, in which they are likely to feel safe. Conversely, contributing to a repository owned by another user may expose developers to less familiar, and potentially less comfortable, environments. To explore this dynamic, we analyze the effect of the analytics dashboard adoption on contribution behavior, distinguishing between contributions to own versus others' repositories.

We repeat our main analyses, but compute all dependent variables separately, conditional on

whether a commit's target repository is owned by the contributing or another user. Panel A in Table 6 reports the results for low-activity users. The regression coefficients align with the direction and significance of our main results. Interestingly, the effect is stronger for contributions to users' own repositories, especially regarding the number of commits, commit message length, and number of unique words. For example, after the adoption, users increase the number of unique words by about 19% for commits to their own repositories, but only by about 5% for commits to others' repositories. The commit message sentiment decreases regardless of repository ownership.

Table 6: Synthetic Difference-in-Differences by Target Repository

	NumCommits		*MesLength*		*MesUniqueWords*		*MesSentiment*	
	(1) Own	(2) Other	(3) Own	(4) Other	(5) Own	(6) Other	(7) Own	(8) Other
	Panel A. Low-Activity Users							
Adoption x After	0.526***	0.077***	0.044***	0.026***	0.188***	0.052***	−0.001***	−0.001**
	(0.004)	(0.002)	(0.003)	(0.006)	(0.006)	(0.011)	(0.000)	(0.001)
Observations	376,704	376,704	184,608	36,936	187,776	37,368	184,608	36,936
Users	31,392	31,392	15,384	3,078	15,648	3,114	15,384	3,078
Mean Adopters Pre	1.596	0.166	35.892	64.428	54.323	120.616	0.001	−0.005
	Panel B. High-Activity Users							
Adoption x After	0.236***	0.096***	0.019***	0.011*	0.017***	0.045***	0.000**	0.002***
	(0.004)	(0.003)	(0.003)	(0.006)	(0.005)	(0.011)	(0.000)	(0.000)
Observations	380,916	380,916	185,760	38,016	189,360	38,376	185,760	38,016
Users	31,743	31,743	15,480	3,168	15,780	3,198	15,480	3,168
Mean Adopters Pre	29.355	9.926	53.974	85.315	255.577	444.083	−0.001	−0.007

Note. The dependent variables are displayed in the first row. The type of repository, whether it is owned by the committing or another user is displayed in the second row. Users are defined as high-activity if the number of commits during the pre-treatment period is equal to or above the sample median. The means of adopters before for *number of commits*, *commit message length*, and *unique words in commit messages* are not logged. The number of users and observations vary between the models because synthDiD requires a balanced panel. Developers can commit without a commit message or the latter may be too short to compute the *commit message length*, the of *unique words in commit messages*, or the *commit message sentiment* partly leading to missing observations and the entire removal of the respective user. Robust standard errors in parentheses. *** $p < 0.01$, ** $p < 0.05$, * $p < 0.1$.

Panel B in Table 6 presents the results for high-activity users, revealing a different pattern compared to low-activity users. While the treatment effect on the number of commits and commit message length is higher in self-owned repositories, there is a sharper increase in the number of unique words and a visibly positive and significant treatment effect on commit message sentiment in contributions to others' repositories.

These findings suggest that the dashboard adoption incentivizes low-activity developers to change their contribution behavior, particularly in their own repositories, while this pattern is less pronounced for more experienced high-activity developers. Low-activity users show greater contribution diversity in their own repositories. In contrast, high-activity users contribute disproportionately more diverse content in others' repositories. This could be because low-activity users—still experimenting with new types of contributions—feel more comfortable in the safe space of their own repositories, while high-activity users are confident enough to contribute more diverse content directly to others' projects. This aligns with our theoretical argument that low-activity users are less experienced and confident in their OSS development skills, potentially lacking important coping mechanisms when confronted with social comparison enabled by the analytics dashboard.

In line with this explanation, high-activity users write more positive commit messages in others' repositories after adoption, possibly indicating that they enjoy the quantification and comparison of their development activities. In contrast, low-activity users tend to write more negative commit messages across all repositories, indicating that the analytics dashboard may have adverse effects on them. While the positive impact for high-activity users aligns with existing research on analytics dashboards (e.g., Berman and Israeli 2022), the negative impact on low-activity users is an important aspect. Recent research has only started to consider these potential costs for analytics users (Wang et al. 2024). Therefore, our focus now shifts to understanding the underlying mechanisms that drive the more negative commit message sentiment among less active users.

6.5 Underlying Mechanism

We hypothesized that adopting the analytics dashboard increases developers' stress, with this effect being more pronounced among initially less active developers. Specifically, we interpret the increasingly negative sentiment of low-activity developers an indicator of stress (Wu et al. 2021).

To better understand whether more negative commit messages indeed reflect stress, we applied

topic modeling to analyze the content of commit messages from less active developers. Because commit messages are typically brief and focused on a single topic, we used Biterm Topic Modeling (BTM) designed for extracting topics from short texts (Yan et al. 2013). BTM has been shown to outperform other topic modeling approaches designed for short texts, such as BERTopic (Miyazaki et al. 2023). More details on our topic modelling approach can be found in Appendix F.

We extracted all commit messages from low-activity users and applied BTM to this corpus. Table 7 presents the results, categorizing the commit messages into eight distinct topics. For better interpretability, we manually named and grouped these topics based on the most frequent words, bigrams, and representative commit messages. The identified topics include four core OSS development activities: bug fixing (topic 1), feature adding (topic 2), code maintenance (i.e., improving working code, topic 3), and testing (topic 4). Four topics encompass peripheral OSS development activities not directly related to coding: version control (i.e., managing different software versions, topic 5), style modifications (e.g., updating the accompanying documentation, topic 6), dependency handling (i.e., ensuring compatibility with changes in underlying packages, topic 7), and branch management (i.e., integrating different versions of the code, topic 8). Next, we calculated the mean commit message sentiment for low-activity users at the user-month-topic level (as opposed to the user-month level used in our main analysis) and ran the same synthDiD analyses as in our main specification, but separately by topic. We were unable to analyze the sentiment for testing (topic 4) due to insufficient complete observations required by synthDiD.

The results of this analysis are presented in Table 8. The DiD estimators show no significant effects on users' commit messages related to bug fixing (topic 1) and feature adding (topic 2), but a positive and significant effect for code maintenance messages (topic 3, $p<0.01$). Thus, the adoption of the analytics dashboard appears to have a neutral to positive impact on commit messages related to core OSS development activities. However, we observe consistently negative DiD estimators for

Table 7: Topics in Low-Activity Users' Commit Messages

ID	Name	Commits	Sentiment	Five most frequent words				
	Core OSS activities							
1	Bug fixing	2,023,121	0.011	fix	use	add	chang	test
2	Feature adding	1,930,339	0.010	add	ad	file	creat	feat
3	Code maintenance	1,746,343	−0.048	fix	test	add	doc	chore
4	Testing	79,860	0.009	test	ad	fix	function	stage
	Peripheral OSS activities							
5	Version control	2,948,944	0.006	de	dev	commit	revert	date
6	Style modification	1,389,900	0.010	fix	add	ad	page	style
7	Dependency handling	1,111,673	0.003	version	use	build	depend	add
8	Branch management	1,085,000	−0.001	bump	request	pull	branch	commit

Note. The table displays the top words per topic as determined by the biterm topic model applied to all commit messages of low activity users considered in the main specification (N = 12,315,180). Topic names are manually assigned based on the most frequent words, bigrams, and exemplary commit messages for each topic. The optimal number of topics (k) is determined through an analysis of associated perplexity and coherence scores for biterm topic models with k = [3,10]. Sentiment is the mean sentiment of all commit messages dealing with the topic.

peripheral OSS development activities. Specifically, after adopting the analytics dashboard less active users write significantly more negative commit messages related to version control (topic 5, p<0.01), style modification (topic 6, p<0.01), and dependency handling (topic 7, p<0.01).

Table 8: Commit Message Sentiment Synthetic Difference-in-Differences by Topics

	Core OSS activities			Peripheral OSS activities			
	(1) Topic 1	(2) Topic 2	(3) Topic 3	(4) Topic 5	(5) Topic 6	(6) Topic 7	(7) Topic 8
Adoption x After	0.000	0.000	0.022***	−0.002***	−0.010***	−0.016***	−0.001
	(0.002)	(0.001)	(0.003)	(0.000)	(0.002)	(0.004)	(0.001)
Observations	20,016	37,440	16,920	85,032	18,072	8,064	9,720
Users	1,668	3,120	1,410	7,086	1,506	672	810
Mean adopters pre	0.009	0.008	−0.066	0.005	0.014	0.003	0.000

Note. The dependent variable is the commit message sentiment separate by topics. The topic identified by Biterm Topic Modelling is displayed in the top row. Only initially less active users are considered in the regressions. Users are defined as low-activity if the number of commits during the pre-treatment period is below the sample median. Robust standard errors in parentheses. *** p < 0.01, ** p < 0.05, * p < 0.1.

Stress is known to manifest in behavioral changes, with stressed individuals tending to be more impatient and negative (Amirkhan et al. 2018, Lunney 2006). Against this backdrop, our findings provide suggestive evidence that the increase in negativity among initially less active developers may indeed be attributed to stress, as these developers may feel that peripheral development

activities distract them from core activities, such as improving their coding skills.

There are two competing mechanisms that could equally explain the observed decrease of the commit message sentiment. First, after adopting the analytics dashboard, the nature of commits made by low-activity users may shift, leading them to work on topics that inherently involve more negative commit messages (e.g., code maintenance, cf., Table 7). Second, users may increasingly commit intermediate steps during their code development process, where the software is not yet fully functional. This could result in more negative phrasing in the associated code documentation.

To investigate these possibilities, we ran regressions segmented by topic across all dependent variables. The results are presented in Appendix G. The analysis reveals a consistent increase in the number of commits across all topics, suggesting that users do not significantly shift their contribution behavior after the adoption of the analytics dashboard. Furthermore, this finding suggests that a rise in intermediate commits is not the primary reason for the decrease in commit message sentiment. If intermediate commits were the driving mechanism, we would expect little to no increase in the number of commits related to peripheral OSS activities, such as updating software documentation. These tasks generally do not involve intermediate steps prone to failure, which could lead to more negative documentation. Instead, they typically reflect a continuous progression of work.

Based on these results, we conclude that negative commit messages among low-activity developers are indeed driven by increased stress. This conclusion is further supported by additional qualitative evidence. We conducted interviews with 32 adopters of the analytics dashboard, including 16 low-activity users. Insights from these interviews corroborate and enrich our interpretation. For example, one respondent, who initially adopted the dashboard as an intern, reported feeling stressed when comparing themselves to more active GitHub developers. However, after transitioning to a professional developer role and gaining confidence in their skills, they no longer experienced

negative emotions from such comparisons. Further details are provided in Appendix H.

7 DISCUSSION

OSS developers' profile pages and the information users choose to display shape social interactions, influencing both their contribution behavior and affect. Our study is among the first to examine this issue. We analyze archival data from 63,135 GitHub developers, one-sixth of whom adopted the *GitHub Readme Stats* analytics dashboard, which publicly displays performance information. Using synthDiD and several robustness checks, our analysis provides evidence that the analytics dashboard positively impacts the number of contributions, contribution diversity, and effort. However, this effect differs between low- and high-activity users. High-activity users appear to benefit from the enabled comparison as they disproportionately alter their behavior in other users' projects when compared to low-activity users. This has two major implications for both developers and the OSS community. First, increased engagement with other projects allows high-activity developers to collaborate more closely with others. This not only helps them further expand their knowledge and experience by learning from fellow developers, but also enables them to enjoy the rewarding experience of close collaboration and exchange in OSS development. Second, the OSS community benefits from the involvement of these high-activity developers, who contribute their valuable resources and expertise to joint projects, strengthening the community that heavily builds on collaborative efforts, as a whole. Conversely, low-activity users primarily adjust their contribution behavior within the "safe space" of their own projects. Furthermore, empirical evidence suggests that initially less active developers write more negative code documentation after adopting the analytics dashboard. An analysis of granular event data provides suggestive evidence that this increase in negativity could be driven by stress among less active developers. Overall, the adoption of the analytics dashboard may widen the gap between low- and high-activity developers, both in terms of coding skills and their social engagement in the community, which could also affect

the satisfaction they derive from it. Additionally, the OSS community as a whole may fail to fully harness the potentially vital contributions of initially low-activity developers to OSS projects.

Our study makes two important contributions to research on OSS developers' contribution behavior. First, prior research has focused on developers' motivation to contribute to OSS, such as identifying social motivators like being an integral part of the community (von Krogh et al. 2012). This shows that the social composition plays a central role in OSS development, also coined "social coding" (Dabbish et al. 2012). We extend this understanding by highlighting the competitive dynamics of social coding, where developers strive to outperform their peers. This competitive drive can be activated by "self-imposed" interventions, such as making performance information publicly visible on prominent spaces like profile pages. Second, while existing studies emphasize the positive aspects of OSS development—such as skill acquisition—there is growing concern about the potential downsides, particularly regarding developers' mental well-being (Tidelift 2023). Building on social comparison theory, we empirically show that upward social comparison can lead to stress, particularly among less active developers. This finding underscores the need to prioritize mental well-being of individual developers and to account for potential heterogeneity among them as an important aspect in future OSS research.

We also contribute to the research on how individuals respond to performance information on online platforms. Previous studies have focused on privately-visible performance information, which often led to positive outcomes such as increased contributions and effort (e.g., Dobrescu et al. 2021, Huang et al. 2019). However, publicly-visible performance information may elicit different responses. Since public information on the platform can act as a performance signal to others, users may feel increased pressure to portray themselves favorable. To our knowledge, our study is the first to empirically examine how public performance information affects individual behavior, revealing potential negative consequences for some users—namely, increased stress. This

suggests that future research should not only focus on instrumental outcomes like the number of contributions or effort but also on the broader implications of publicly-visible performance information. Specifically, future studies in the field should balance attention between the individual user experience and the overall welfare of the community. These two perspectives may interact in complex ways. For instance, early-stage developers who experience stress from constant negative social comparison may temporarily increase their contributions but drop out of the community in the long term. Subsequent research could explore strategies for OSS platforms and communities to recognize the unique contributions of each member, regardless of their background.

Finally, our findings contribute to the literature on the effects of analytics. Analytics have become an integral part of everyday life, influencing business decisions and even leisure activities, for example, by tracking workout performance (Berman and Israeli 2022, Bojd et al. 2022). Previous research has demonstrated that analytics can drive performance gains, yielding monetary benefits for organizations and improved workout outcomes for individuals (Berman and Israeli 2022, Müller et al. 2018, Bojd et al. 2022). Our study corroborates these findings at the individual level, showing that analytics-enabled comparisons motivate individuals to exert greater effort. However, recent research also points to potential negative effects of analytics for consumers, such as higher prices when sellers in high-demand markets leverage analytics for pricing strategies (Wang et al. 2024). Our research extends this narrative by showing that negative effects can also occur at the user level, particularly by widening the gap between low- and high-activity users and increasing stress for low-activity individuals. Taken together, these insights suggest that future research investigating the effects of analytics should adopt a broader conceptualization of affected stakeholders. Specifically, the findings highlight that not only analytics-adopting organizations are impacted, but also individuals and others they interact with.

Our study has limitations that future research should address. The empirical analysis relies

on available archival data from GitHub, which lacks detailed information on the quality of code contributions or developers' professional backgrounds. Future research could delve into these aspects through field experiments, offering richer insights into the analytics impact on developers and their contributions. Specifically, it might be possible to obtain data on developers' professional backgrounds and current situations, such as job searches or the relevance of OSS development in their daily work. Additionally, researchers could approximate the quality of code contributions through more in-depth analysis of developer activity and potential spillover effects of analytics adoption on their networks. Importantly, field experiments would enable developers to engage in real-world settings, providing a more accurate reflection of social interactions and comparisons that are an integral part of OSS development.

Finally, our research has implications for OSS platforms and developers. By enabling social comparison, OSS platforms can incentivize developers to contribute more, with greater effort and diversity. For platforms or developers aiming to increase code contributions, implementing an artifact like the analytics dashboard could be a promising low-cost approach. The benefits of the analytics dashboard extend beyond its direct users. Developers who do not actively use the dashboard may also gain advantages, as the increased effort and diversity in adopters' contributions result in longer and more detailed software code documentation. This improved documentation enhances the ability of other developers to track adopters' progress and potentially learn from their work. Moreover, this improvement in documentation may positively impact the size and traceability of training data for large language models, which is constituted by openly accessible data such as GitHub contributions. However, OSS platforms and developers must make this decision consciously and be aware that enabling social comparison may induce stress among less active developers. Over time, stress may not only harm individual developers but could also affect the platform, as the resulting emotional strain can reduce developers' willingness to contribute (Shankar et al. 2024).

8 CONCLUSION

OSS developers' contribution behavior is a social phenomenon that revolves around the information displayed on developers' public profile pages. However, prior literature has not sufficiently explored how the presented information influences OSS developers' contributions and affect. We address this gap by investigating the effects of adopting a popular analytics dashboard on a developer's profile page. In our large-scale empirical study, we find that the adoption is a double-edged sword—developers contribute more but this comes at the cost of increased stress induced by excessive upward social comparison for some developers. These findings carry implications for practitioners and researchers. Most importantly, interventions that influence OSS developers' contribution behavior may have unexpected but decisive adverse effects.

NOTES

[1]For example, intense discussions revolved around the removal of a streak counter from user profiles that displayed the number of consecutive days a user contributed to GitHub: https://github.com/isaacs/github/issues/627 and https://github.com/dear-github/dear-github/issues/163 (Retrieved January 2, 2025).

[2]https://github.com/anuraghazra/github-readme-stats/pull/960 (Retrieved January 2, 2025).

[3]Measuring the effort put into a contribution by its documentation length is a common proxy that has proven effective in comparable online contexts (cf., Burtch et al. 2022, Pethig et al. 2024).

[4]To make it computationally feasible for the dependent variable *NumCommits*, the sample size of non-adopters is reduced to **twice** the relevant adopters and the repetitions to **five**.

ACKNOWLEDGEMENTS

This work was supported by the University of Mannheim's Graduate School of Economic and Social Sciences. Furthermore, the authors acknowledge support by the state of Baden-Wuerttemberg

through bwHPC and the Julius Paul Stiegler Memorial Foundation that supported a conference visit where a previous version of this paper was presented.

REFERENCES

Amirkhan JH, Landa I, Huff S (2018) Seeking signs of stress overload: Symptoms and behaviors. *International Journal of Stress Management* 25(3):301–311.

Angrist JD, Pischke JS (2008) *Mostly Harmless Econometrics: An Empiricist's Companion* (Princeton, NJ, USA: Princeton University Press).

Arkhangelsky D, Athey S, Hirshberg DA, Imbens GW, Wager S (2021) Synthetic difference-in-differences. *American Economic Review* 111(12):4088–4118.

Berman R, Israeli A (2022) The value of descriptive analytics: Evidence from online retailers. *Marketing Science* 41(6):1074–1096.

Bojd B, Song X, Tan Y, Yan X (2022) Gamified challenges in online weight-loss communities. *Information Systems Research* 33(2):718–736.

Burtch G, Hong Y, Bapna R, Griskevicius V (2018) Stimulating online reviews by combining financial incentives and social norms. *Management Science* 64(5):2065–2082.

Burtch G, He Q, Hong Y, Lee D (2022) How do peer awards motivate creative content? Experimental evidence from Reddit. *Management Science* 68(5):3488–3506.

Buunk AP, Gibbons FX (2007) Social comparison: The end of a theory and the emergence of a field. *Organizational Behavior and Human Decision Processes* 102(1):3–21.

Califf C, Sarker S, Sarker S (2020) The bright and dark sides of technostress: A mixed-methods study involving healthcare IT. *MIS Quarterly* 44(2):809–856.

Chan R (2022) Open-source developers are burning out, quitting, and even sabotaging their own projects — and it's putting the entire internet at risk. *Business Insider*. Retrieved January 12, 2023, https://www.businessinsider.com/open-source-developers-burnout-low-pay-internet-2022-3.

Chen Y, Harper FM, Konstan J, Li SX (2010) Social comparisons and contributions to online communities: A field experiment on MovieLens. *American Economic Review* 100(4):1358–1398.

Dabbish L, Stuart C, Tsay J, Herbsleb J (2012) Social coding in GitHub: Transparency and collaboration in an open software repository. *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, 1277–1286.

Daniel LM, Maruping L, Cataldo M, Herbsleb J (2018) The impact of ideology misfit on open source software communities and companies. *MIS Quarterly* 42(4):1069–1096.

de Vries DA, Kühne R (2015) Facebook and self-perception: Individual susceptibility to negative social comparison on Facebook. *Personality and Individual Differences* 86:217–221.

Devlin J, Chang MW, Lee K, Toutanova K (2019) BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv*. Retrieved January 19, 2023, http://arxiv.org/abs/1810.04805.

Dissanayake I, Zhang J, Yasar M, Nerur SP (2018) Strategic effort allocation in online innovation tournaments. *Information & Management* 55(3):396–406.

Dobrescu LI, Faravelli M, Megalokonomou R, Motta A (2021) Relative performance feedback in education: Evidence from a randomised controlled trial. *The Economic Journal* 131(640):3145–3181.

Festinger L (1954) A theory of social comparison processes. *Human Relations* 7(2):117–140.

Gerber JP, Wheeler L, Suls J (2018) A social comparison theory meta-analysis 60+ years on. *Psychological Bulletin* 144(2):177–197.

Gibbons FX, Lane DJ, Gerrard M, Reis-Bergan M, Lautrup CL, Pexa NA, Blanton H (2002) Comparison-level preferences after performance: Is downward comparison theory still useful? *Journal of Personality and Social Psychology* 83(4):865–880.

GitHub Inc (2022) GitHub glossary. Retrieved January 16, 2023, https://docs.github.com.

GitHub Inc (2023) GitHub about. Retrieved February 02, 2023, https://github.com/about.

Guzman E, Azócar D, Li Y (2014) Sentiment analysis of commit comments in github: An empirical study. *Proceedings of the 11th Working Conference on Mining Software Repositories*, 352–355.

Hattori LP, Lanza M (2008) On the nature of commits. *2008 23rd IEEE/ACM International Conference on Automated Software Engineering - Workshops*, 63–71.

Hauff C, Gousios G (2015) Matching GitHub developer profiles to job advertisements. *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, 362–366.

Hazra A (2020) GitHub Readme Stats. Retrieved March 5, 2023, https://github.com/anuraghazra/github-readme-stats.

Hoffmann M, Nagle F, Zhou Y (2024) The value of open source software. *SSRN*. Retrieved January 19, 2024, https://papers.ssrn.com/abstract=4693148.

Huang N, Burtch G, Gu B, Hong Y, Liang C, Wang K, Fu D, Yang B (2019) Motivating user-generated content with performance feedback: Evidence from randomized field experiments. *Management Science* 65(1):327–345.

Huang N, Zhang J, Burtch G, Li X, Chen P (2021) Combating procrastination on massive online open courses via optimal calls to action. *Information Systems Research* 32(2):301–317.

Johnson SL, Faraj S, Kudaravalli S (2014) Emergence of power laws in online communities: The role of social mechanisms and preferential attachment. *MIS Quarterly* 38(3):795–808.

Kalliamvakou E, Gousios G, Blincoe K, Singer L, German DM, Damian D (2016) An in-depth study of the promises and perils of mining GitHub. *Empirical Software Engineering* 21(5):2035–2071.

Krasnova H, Widjaja T, Buxmann P, Wenninger H, Benbasat I (2015) Research note—Why following friends can hurt you: An exploratory investigation of the effects of envy on social networking sites among college-age users. *Information Systems Research* 26(3):585–605.

Landers RN, Bauer KN, Callan RC (2017) Gamification of task performance with leaderboards: A goal setting experiment. *Computers in Human Behavior* 71:508–515.

Lee SY (2014) How do people compare themselves with others on social network sites?: The case of Facebook. *Computers in Human Behavior* 32:253–260.

Lindberg A, Schecter A, Berente N, Hennel P, Lyytinen K (2024) The entrainment of task allocation and release cycles in open source software development. *MIS Quarterly* 48(1):67–94.

Lu L, Yuan YC, McLeod PL (2012) Twenty-five years of hidden profiles in group decision making: A meta-analysis. *Personality and Social Psychology Review* 16(1):54–75.

Lumbard K, Germonprez M, Goggins S (2024) An empirical investigation of social comparison and open source community health. *Information Systems Journal* 34(2):499–532.

Lunney M (2006) Stress overload: A new diagnosis. *International Journal of Nursing Terminologies and Classifications* 17(4):165–175.

Marlow J, Dabbish L, Herbsleb J (2013) Impression formation in online peer production: Activity traces and personal profiles in Github. *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, 117–128.

Maruping LM, Daniel SL, Cataldo M (2019) Developer centrality and the impact of value congruence and incongruence on commitment and code contribution activity in open source software communities. *MIS Quarterly* 43(3):951–976.

Matook S, Lee G, Fitzgerald B (2021) Information systems development. Retrieved January 13, 2022, https://www.misqresearchcurations.org/blog/2021/12/7/information-systems-development.

Medappa PK, Srivastava SC (2019) Does superposition influence the success of FLOSS projects? An examination of open-source software development by organizations and individuals. *Information Systems Research* 30(3):764–786.

Miyazaki K, Murayama T, Uchiba T, An J, Kwak H (2023) Public perception of generative AI on twitter: An empirical study based on occupation and usage. *arXiv*. Retrieved July 14, 2023, http://arxiv.org/abs/2305.09537.

Müller O, Fay M, Vom Brocke J (2018) The effect of big data and analytics on firm performance: An econometric analysis considering industry characteristics. *Journal of Management Information Systems* 35(2):488–509.

Oberfichtner M, Tauchmann H (2021) Stacked linear regression analysis to facilitate testing of hypotheses across OLS regressions. *The Stata Journal* 21(2):411–429.

Pethig F, Hoehle H, Hui KL, Lanz A (2024) Behavior toward newcomers and contributions to online communities. *MIS Quarterly*. Forthcoming.

Raman N, Cao M, Tsvetkov Y, Kästner C, Vasilescu B (2020) Stress and burnout in open source: Toward finding, understanding, and mitigating unhealthy interactions. *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: New Ideas and Emerging Results*, 57–60.

Shankar R, Wang L, Gunasti K, Li H (2024) Nonverbal peer feedback and user contribution in online forums: Experimental evidence of the role of attribution and emotions. *Journal of the Association for Information Systems* 25(2):267–303.

Shi L, Huang P, Ramaprasad J (2021) Pragmatic men, romantic women: Popularity feedback on online dating platforms. *SSRN*. Retrieved November 5, 2024, https://papers.ssrn.com/abstract=3967702.

Smirnova I, Reitzig M, Alexy O (2022) What makes the right OSS contributor tick? Treatments to motivate high-skilled developers. *Research Policy* 51(1):104368.

Tidelift (2023) The 2023 Tidelift state of the open source maintainer report. Technical report.

von Krogh G, Haefliger S, Spaeth S, Wallin MW (2012) Carrots and rainbows: Motivation and social practice in open source software development. *MIS Quarterly* 36(2):649–676.

Wang Z, Guo H, Liu D (2024) Economics of analytics services on a marketplace platform. *MIS Quarterly* 48(2):775–823.

White K, Lehman DR (2005) Culture and social comparison seeking: The role of self-motives. *Personality and Social Psychology Bulletin* 31(2):232–242.

Wright NL, Nagle F, Greenstein S (2024) Contributing to growth? The role of open source software for global startups. *SSRN*. Retrieved January 31, 2024, https://papers.ssrn.com/abstract=4699182.

Wu J, Ye C, Zhou H (2021) BERT for sentiment classification in software engineering. *2021 International Conference on Service Science*, 115–121.

Yan X, Guo J, Lan Y, Cheng X (2013) A biterm topic model for short texts. *Proceedings of the 22nd International Conference on World Wide Web*, 1445–1456.

Yeverechyahu D, Mayya R, Oestreicher-Singer G (2024) The impact of large language models on open-source innovation: Evidence from GitHub Copilot.

Yue Z, Zhang R, Xiao J (2022) Passive social media use and psychological well-being during the COVID-19 pandemic: The role of social comparison and emotion regulation. *Computers in Human Behavior* 127:107050.

Zhang C, Hahn J, De P (2013) Research note—Continued participation in online innovation communities: Does community response matter equally for everyone? *Information Systems Research* 24(4):1112–1130.